US005128945A

United States Patent [19]

Enns et al.

[11] Patent Number: 5,128,945

[45] Date of Patent: Jul. 7, 1992

[54] PACKET FRAMING USING CYCLIC REDUNDANCY CHECKING

[75] Inventors: Frederick Enns, Menlo Park; J. Paul O'Hare, Felton, both of Calif.

[73] Assignee: Stratacom, Inc., San Jose, Calif.

[21] Appl. No.: 786,016

[22] Filed: Oct. 31, 1991

Related U.S. Application Data

[63] Continuation of Ser. No. 481,709, Feb. 15, 1990, Pat. No. 5,072,449, which is a continuation-in-part of Ser. No. 454,258, Dec. 21, 1989, abandoned.

[51] Int. Cl.[5] G06F 11/10; H04J 3/24

[52] U.S. Cl. 371/37.1; 370/94.2; 370/105.1; 375/108

[58] Field of Search 371/37.1, 47.1, 46, 371/42, 53, 54, 20.1; 370/105, 105.1, 106, 100.1, 94.1, 94.2, 14; 375/108, 114, 116

[56] References Cited

U.S. PATENT DOCUMENTS

4,316,285 2/1982 Bobilin ..... 371/47.1 X

4,914,654 4/1990 Matsuda 370/94.1

4,947,484 8/1990 Twitty 371/37.1

4,962,498 10/1990 May 370/94.1

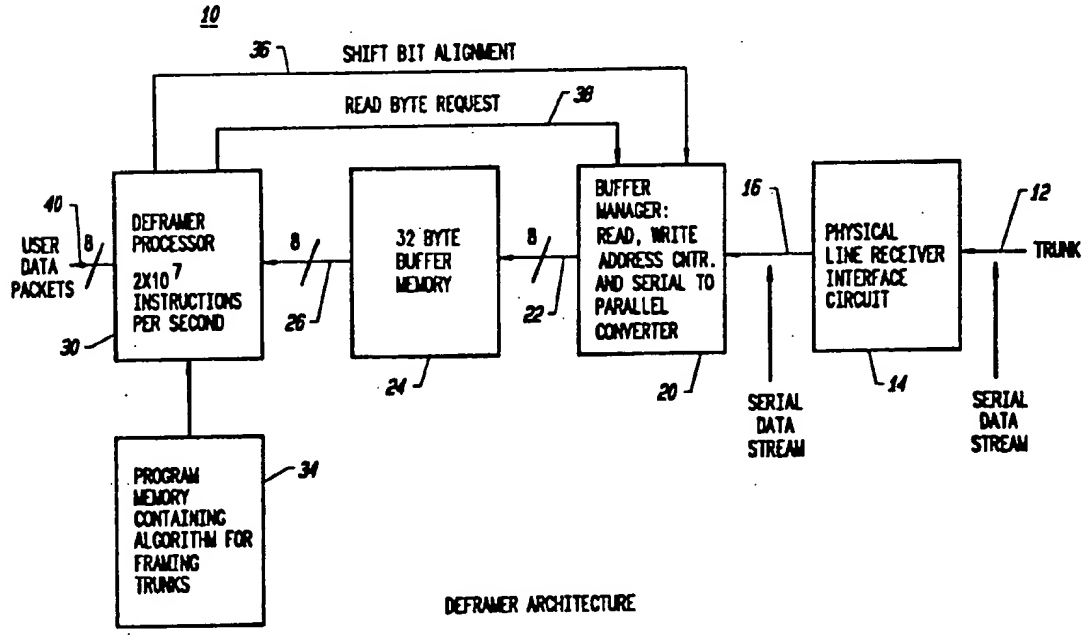
Primary Examiner—Robert W. Beausoliel

Attorney, Agent, or Firm—Flehr, Hohbach, Test, Albritton & Herbert

[57] ABSTRACT

Packets transported over a serial data stream need to have their start and stop information encoded so that they can be parsed by the receiving device. Techniques used in various standards require extra bandwidth to carry this information. For framed data streams like T1, proprietary protocols have used the frame structure to mark packet boundaries. Both types of solutions are desirable for unframed, long distance data streams like the CCITT G.703 2.048 Mbps unframed service, subrate trunks, or fractional T1 services. The present invention provides a packet framing method that uses the Cyclic Redundancy Check (CRC) found in most packet formats. The protocol does not require any extra bandwidth and can work with any serial data stream physical layer.

12 Claims, 5 Drawing Sheets



DEFRAMER ARCHITECTURE

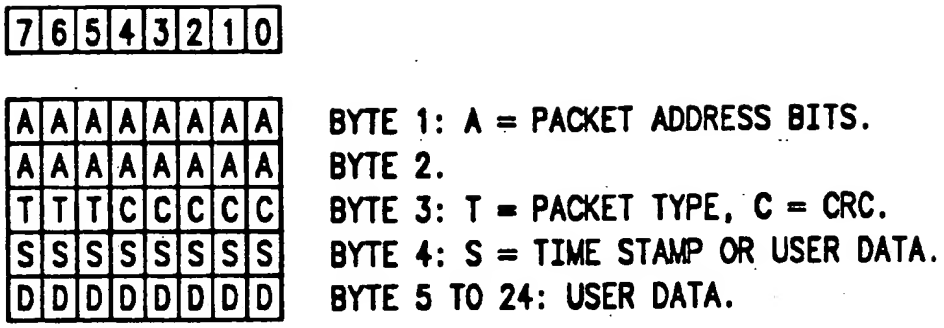


FIG. 1

IPX PACKET FORMAT

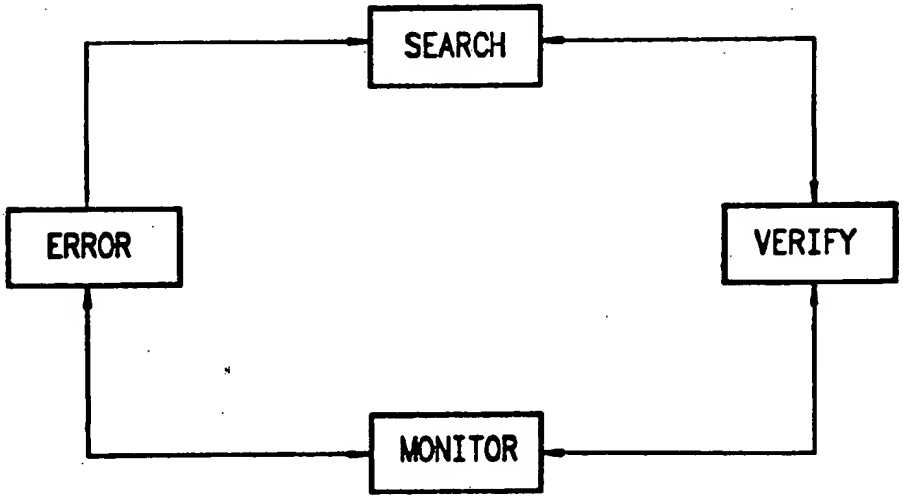


FIG. 2

FRAMING ALGORITHM STATE MACHINE

	E1 2.048Mbps	64Kbps	56Kbps
WORST CASE BUSY LINE	5.9	188	215
TYPICAL BUSY LINE	2.4	77	87
WORST CASE IDLE LINE	4.1	130	149
TYPICAL IDLE LINE	1.5	48	54

TABLE 1: PACKET FRAMING TIMES IN MSEC FOR TRUNKS WITHOUT BYTE ALIGNMENT

FIG. 3

	E1 1.984Mbps	T1 1.536Kbps	(FRAC. T1) 64Kbps
WORST CASE BUSY LINE	0.98	1.2	30
TYPICAL BUSY LINE	0.61	0.79	18.9
WORST CASE IDLE LINE	0.45	0.58	14
TYPICAL IDLE LINE	0.26	0.33	8.0

TABLE 2: PACKET FRAMING TIMES IN MSEC FOR TRUNKS WITH BYTE ALIGNMENT

FIG. 4

	CYCLES USED PER 125 USEC.	PERCENT UTILIZATION
SEARCH MODE	1660	66%
MONITOR MODE	1200	48%

2500 INSTRUCTIONS PER 125 USEC PERIOD

TABLE 3: NTC DEFRAMER PROCESSOR UTILIZATION

FIG. 5

	CYCLES USED PER 125 USEC.	PERCENT UTILIZATION
TRANSMIT MODE	1660	66%

2500 INSTRUCTIONS PER 125 USEC PERIOD

TABLE 3: NTC FRAMER PROCESSOR UTILIZATION

FIG. 6

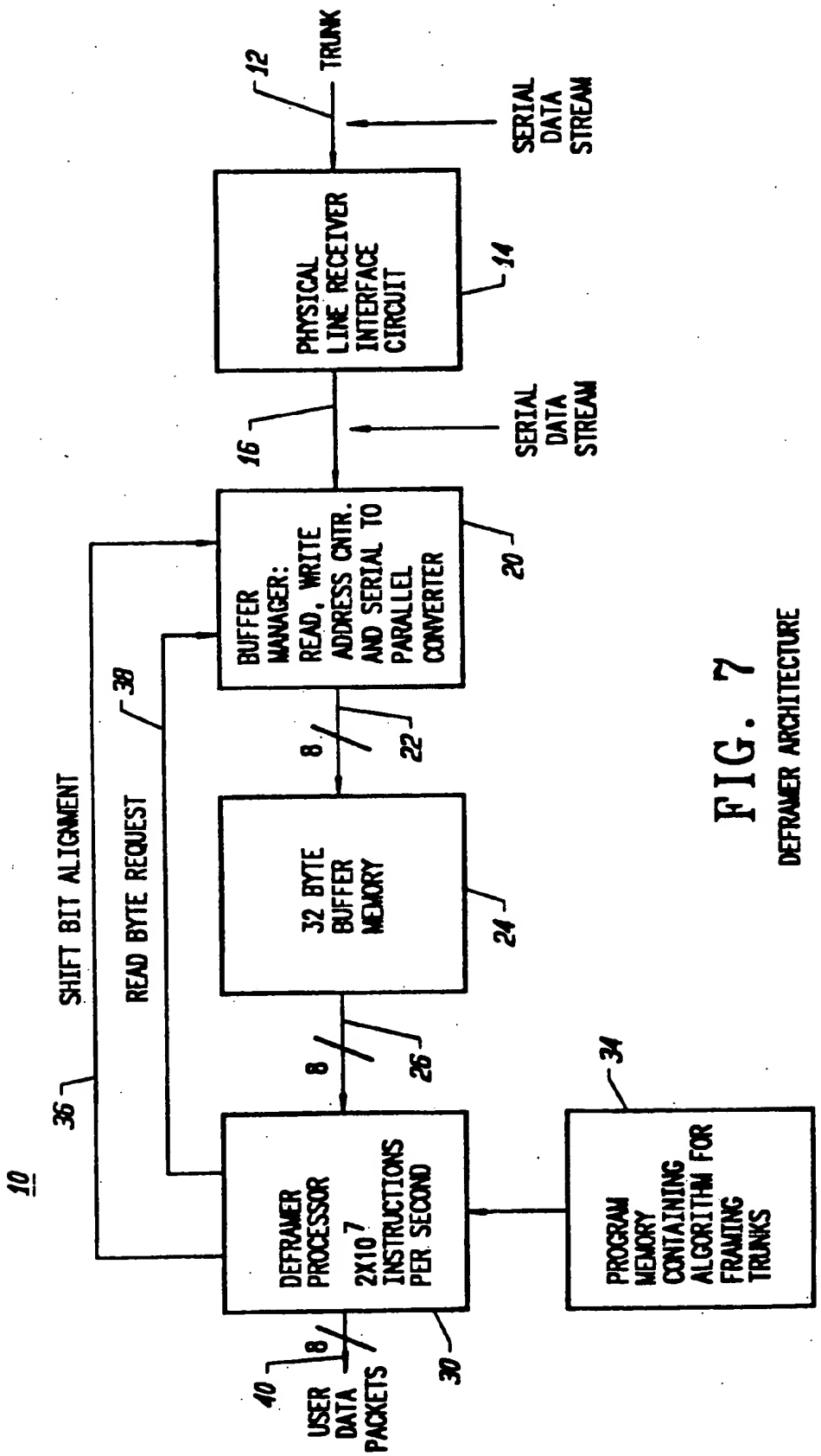


FIG. 7
DEFRAMER ARCHITECTURE

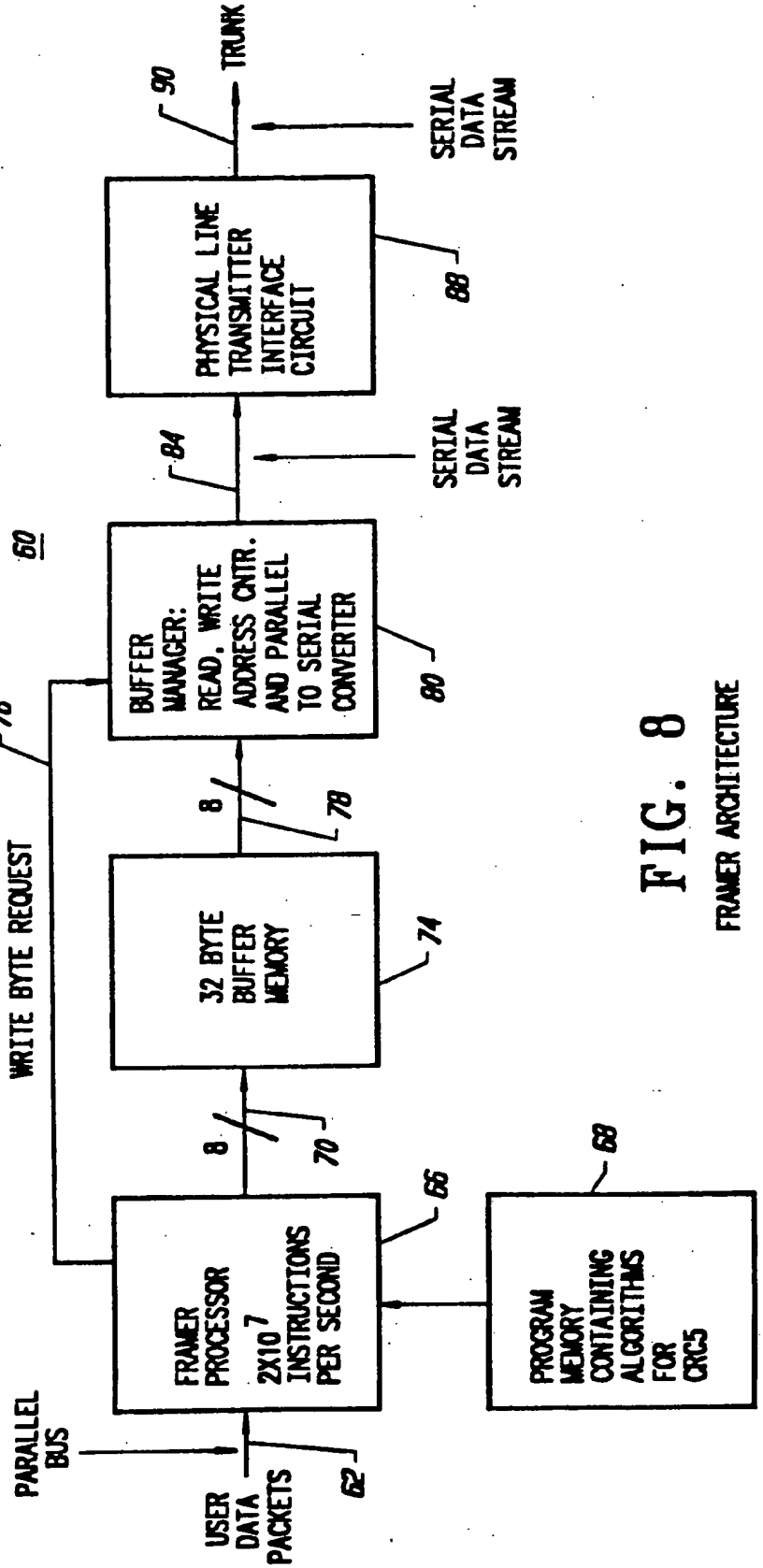


FIG. 8
FRAMER ARCHITECTURE

PACKET FRAMING USING CYCLIC REDUNDANCY CHECKING

This is a continuation of application Ser. No. 07/481,709 filed Feb. 15, 1990, which is a continuation-in-part of Ser. No. 07/454,258, filed Dec. 21, 1989 now abandoned.

BACKGROUND OF THE INVENTION

The present invention relates to a data communications system and method utilizing packet framing techniques.

Serial data streams that use packets protocols to transfer information need to establish byte alignment of the data and mark the beginning and end of the packets.

In the past various methods have been used to accomplish these tasks. HDLC (High-level Data Line Control) protocols use a unique eight bit flag to mark a packet's start and end. Inside the packet the data is bit stuffed to prevent an accidental flag from being sent. The 802.3 "Ethernet" standard is typical of another set of protocols which use an idle preamble followed by a non-idle packet start header to mark the beginning of a packet.

These methods have worked well in applications where band width efficiency has not been an issue. However in telecommunications, the customer typically rents bandwidth from a service provider. Protocols for this application have to maximize the bandwidth available for the customer's data to save money on the purchased service. The framing algorithm should have as low an overhead as possible. For example if 24 bytes of user data were carried in an HDLC frames packet, it could have as much as 27 bits of data stuffed into it to prevent a false flag.

To this would be added a start and end flag of eight bits each. Such a worst case packet would have a 22% packet framing overhead. For a T1 line which can cost hundreds or thousands of dollars a month, this is expensive.

StrataCom, Inc. manufactures a digital multiplexer (known as the Integrated Packet Exchange (IPX)) that connects customer's voice and data equipment together over T1 lines using a packet transport protocol. That IPX multiplier is described in U.S. Pat. No. 4,771,425, the details of which are hereby incorporated by reference.

Presently the IPX solves the problem of framing packets on T1 lines by making all of its packets 24 bytes long and placing them inside a T1 frame. The T1 frame format has one framing bit and 24 bytes of data; 193 bits total. The T1 framing bit is required for the line to use T1 vendor services. Therefore, this packet framing, which is the same as the same as the T1 line framing, needs no extra bandwidth.

This solution does not work with other types of trunk interfaces and for packets that are not 24 bytes long. Fractional T1 and framed E1 (E1 is the 2.048 Mbps trunk defined by CCITT G.703 standard) can supply byte alignment but do not have 24 bytes per frame for the packet alignment needed by the IPX packet protocol. Unframed E1 (Britain's Mega-Stream), V.11 (Britain's Kilo-Stream and France's Transfix) and V.35 subrate trunks do not have either byte or packet alignment signals in their physical interface.

SUMMARY OF THE INVENTION

It is an object of the present invention to provide an improved data communication system and method utilizing packet framing techniques.

It is a more particular object of the present invention to provide packet framing using cyclic redundancy checking.

The CRC framing algorithm described below is a protocol that byte aligns and packet frames a serial bit stream from the existing data structure in common packet formats. It achieves packet framing without additional bandwidth overhead.

A solution to the packet framing problem must use as little bandwidth as possible, be transparent to the user's data, accommodate high packet throughput rate, and be robust. By robust, it is meant that the algorithm must be fast in recovering from errors, it must achieve quick packet alignment (T1 synchronization times of 10 msec are acceptable), and it should have low error multiplication (a transmission error affecting the framing algorithm should not cause a large amount of data to be lost).

Other objects, features and advantages of the present invention will become apparent from the following detailed description when taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings which are incorporated in and form a part of this specification illustrate an embodiment of the invention and, together with the description, serve to explain the principles of the invention.

FIG. 1 depicts an IPX packet format for describing the operation of the present invention.

FIG. 2 depicts a framing algorithm state machine for describing the operation of the present invention.

FIG. 3 depicts a table illustrating packet framing time for trunks without byte alignment.

FIG. 4 depicts a table showing packet framing times for trunks with byte alignment.

FIG. 5 depicts a table illustrating deframer processor utilization.

FIG. 6 depicts a table illustrating framer processor utilization.

FIG. 7 depicts deframer architecture utilized with the present invention.

FIG. 8 depicts framer architecture utilized with the present invention.

DETAILED DESCRIPTION OF THE DRAWINGS

Reference will now be made in detail to the preferred embodiment of the invention, an example of which is illustrated in the accompanying drawings. While the invention will be described in conjunction with the preferred embodiment, it will be understood that it is not intended to limit the invention to that embodiment. On the contrary, it is intended to cover alternatives, modifications and equivalents as may be included within the spirit and scope of the invention as defined by the appended claims.

PROTOCOL DESCRIPTION

The CRC packet framing technique can be applied to any packet format that has an error checking code embedded in it. Described here is the way packet framing

is achieved using one preferred packet format with its CRC code.

Referring now to FIG. 1, the IPX packet format is a fixed 24 byte packet that contains either a three or four byte header depending on the packet type, see FIG. 1. The first two bytes (Bytes 1 and 2) are the packet's destination address. The third byte contains a three bit packet type identification number and a five bit Cyclic Redundancy Check sum. The fourth byte is used only by certain types of packets; it contains a time stamp. The CRC is calculated over the first four bytes of the packet regardless of whether the packet type has a time stamp byte or not.

A framing algorithm state machine is shown in FIG. 2. The CRC framing algorithm has four states. The machine initializes itself in the "Search" state. The "Search" state looks for a good CRC in the serial bit stream. When a good CRC is found the algorithm enters a "Verify" state where it checks consecutive packets for good CRC to make sure that the packet frame boundary has been found. If there is a bad CRC in the "Verify" state, the algorithm returns to the "Search" state.

In the "Verify" state a packet is good if its CRC and the CRC of the next packet are correct. If the verification is successful then the "Monitor" state is entered. In this state the algorithm enables the receiver to store packets and it continues to check the received packet's CRC. If a bad CRC is found the algorithm goes to an "Error" state. In the "Error" state, good packets continue to be received and the CRC checked. A packet framed by good CRSs immediately returns the algorithm to "Monitor" state; consecutive bad CRC checks puts the algorithm in the "Search" state.

In the "Search" state the CRC framing algorithm looks at the serial bit stream for bit sequences that have a valid check sum. In the case for the IPX format, the algorithm looks for a valid CRC(5) over a 32 bit span of data.

The specific IPX framing implementation is executed by a micro processor and a hardware receive buffer that can be bit aligned. The framing architecture is shown in FIG. 8 and the deframer architecture is shown in FIG. 7.

The buffer is one byte wide and it receives data directly from the lines interface chips. The processor can control the bit alignment within the byte of the data put into the buffer of the interface. The framing algorithm is optimized so that the processor's real time requirements in the search phase are kept low.

This is done by having the processor scan a window of 32 bits from the hardware buffer for good CRC and then discharging the data if the check fails. The processor is freed from the bit manipulations of the data that would be required if it were to look for all the possible patterns within a window of data. The time required to find a packet frame boundary is increased because data is discarded, but doing so makes the real time requirements of the processor easier to estimate and engineer. On a given byte boundary, the processor checks six windows of 32 bits each for two good CRC's framing a packet. If no match is found, the processor shifts the bit alignment of the hardware buffer. The CRC is then checked over another six windows (24 bytes) of data. The algorithm continues bit shifting until a good CRC is found. Two good CRSs framing a packet changes the algorithm's state from "Search" to "Verify".

The "Verify" mode is used to insure that an accidental CRC match is not used to frame the packet line. For the IPX the CRC(5) has a probability of one chance in thirty two of finding a good CRC in random data. The "Verify" state requires four packets in a row be framed by five good CRCs to frame synchronize the line. For random data the probability of a false frame is $3*10^{-8}$. No packets are accepted by the receiver state machine in the "Verify" state.

Packet boundaries used to calculate CRCs in the "Verify", "Monitor", and "Error" states are determined by the packet length. Normal IPX packets have a fixed, 24 byte length. A special four byte long "Idle Trunk Packet" is used between IPX nodes when no user data packets are available to send. The "Idle" packet's shorter length helps speed the packet framing processes. A special IPX packet address (see FIG. 1) is used to identify the "Idle" packet.

When a good CRC sequence is found in the "Verify" state the line is considered to be synchronized to both the data stream's byte boundary and to the packet frame. The framing algorithm then goes to the "Monitor" state and packet reception starts.

If in the "Monitor" state a bad CRC is found, caused by either a transmission error or a loss of synchronization, then the "Error" state is entered. The "Error" state checks the packet framing synchronization without throwing out good packets for shifting the frame byte boundary. This prevents a simple line error from affecting more than one packet (error multiplication). For the IPX, desynchronization is not started until forty eight bytes of data are checked and no good, CRC framed packet is found.

A pseudo code implementation of the IPX algorithm is given in Appendix A.

The algorithm is flexible. It can be applied to more than the IPX packet protocol. The CRC can be different sizes (ie. CRC(8), CRC(16)); the CRC can cover some or all of a packet; and the packets can have a variable length. For variable length packets, the length field can be read to know where the next packet should start.

If the trunk has its own byte framing built in like T1 or framed E1, the CRC framing algorithm saves time by only doing byte shifts of the data stream to find the packet boundaries.

CRC Packet Framing Performance for the IPX

Bandwidth:

No extra bandwidth required; the framing is done with the existing packet's CRC.

CRC(5) Characteristics:

The CRC(5) used by the IPX is five bits long, cover 32 bits (including itself), and uses the polynomial: $X^5+X^4+X^3+X^2+X^0=61$ (decimal)—

The error coverage for 32 bits is:

100% of single bit errors.

97% of double bit errors.

100% of bursts less than 5 bits long.

97% of random data patterns.

Time to Frame a Line:

The time to packet frame a line depends on the data rate of the line, whether the line's data is byte aligned or not, the nature of the data pattern on the line, and how many "Idle" packets are being sent. Four types of times are calculated; worst case busy, worst case idle, typical busy, and typical idle.

The "Worst Case" times assume the worst bit and byte alignment requiring the most shifts and searches. They also assume that all the data is random. The number of false CRC's seen in one stan-

dard deviation greater than the mean expected number. The worst case probabilities in Table 1 are 0.2%, in Table 2 they are 2%.

The "Typical Case" times use average numbers for the number of bit and byte shifts and for the number of false CRCs.

All the T1 and E1 trunk packet framing times compare favorably with T1 line framing times in the 10 msec range.

Error Multiplication:

When an error is detected in the "Monitor" state, the IPX packet receiver drops that packet and goes to the "Error" state. Good CRC packets in the "Error" state are transmitted on the IPX bux. For a bit error in the packet header, error multiplication is limited to loosing one packet of up to 21 bytes of customer data. This is the same error multiplication as in the current IPX product without CRC packet framing.

Robustness:

The packet framing algorithm is inherently robust. The "Verify" state checks that the line has not falsely framed to an erroneous packet boundary. The probability of a false frame is 3×10^{-8}. Even if a line were to falsely frame, the "Monitor" state constantly checks for loss of packet frame synchronization. In such a circumstance the "Error" state would re-initiate the packet frame search.

Packet Throughput:

The IPX Network Trunk Card (NTC) implements the CRC packet framing algorithm with two processors; a Framer (transmit to the line) and a Deframer (receive from the line). The real time required for each processor has been estimated for their complete set of tasks which include functions in addition to CRC framing and deframing. The numbers in Tables 3 and 4 are worst case averages of processor utilization based on simulations of the code and data. Each processor has 2500 instruction cycles available to it in a 125 micro-second period.

Data Transparency:

The CRC packet framing algorithm places no constraints on the user's data. False CRC matches within the data do not permanently keep the algorithm from finding the correct packet frame alignment.

Referring now to FIG. 7, deframer architecture **10** according to the present invention is shown, in which serial data stream via trunk **12** is input to a physical line receiver interface circuit **14**. The output of circuit **14** is a serial data stream on bus **16** which is input to a buffer manager **20**.

Buffer manager **20** includes a read/write address counter and serial to parallel converter for outputting a 8-bit data on bus **22** to 32-byte buffer memory **24**.

The output of memory **24** on 8-bit bus **26** is input to a deframer processor **30** which processes 2×10^7 instructions per second.

The deframer processor **30** provides a shift bit alignment signal on lead **36** to buffer manager **20**. Also, deframer processor **30** provides a read/write request signal on lead **38** to buffer manager **20**.

Deframer processor **30** receives control information from a program memory **34**. The control information contains an algorithm for framing the trunks. The output of deframer processor **30** is 8-bit bus **40** carrying user data packets.

Fig. **8** depicts a framer architecture **60** according to the present invention. In FIG. **8**, the framer architecture includes a frame processor **66** which receives user data packets on parallel bus **62**. Frame processor **66** processes 2×10^7 instructions per second.

The output of framer processor **66** is via 8-bit bus **70** to 32-byte buffer memory **74**.

The output of memory **74** is via 8-bit bus **78** to a buffer manager **80** which includes a read/write address counter and parallel to serial converter.

Framer processor provides a write byte request signal on lead **76** to buffer manager **80**. Framer processor receives instructions from program memory **68** containing control algorithms for a cyclic redundancy check according to the present invention.

The output of buffer manager **80** is a serial data stream on lead **84** which is connected to physical line transmitter interface circuit **88** for output as a serial data stream to trunk **90**.

The foregoing description of the preferred embodiment of the invention has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed, and many modifications and variations are possible in light of the above teaching. For example, error detection information in general could be utilized with the present invention. The preferred embodiment utilizing cyclic redundancy checking was chosen and described in order to best explain the principles of the invention and its practical applications to thereby enable others skilled in the art to best utilize the invention and various embodiments and with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined only by the claims appended hereto.

Appendix A: The IPX Pseudo Code Packet Framing Algorithm

```
DEFRAMER ()
  {
   STATE = SEARCH;                    /* INITIAL CONDITION */
   IF (LINE = T1 OR E1_31_CHANNEL OR E1_30_CHANNEL OR
      FRAC_T1 THEN SET_LINE_TYPE = FRAMED);
   ELSE (LINE_TYPE = UNFRAMED);
   WHILE (NON-STOP)
     {
      SWITCH (STATE)
        {
         CASE SEARCH:  SEARCH ();
                       BREAK;
```

```
            CASE VERIFY:  VERIFY ();
                          BREAK;
            CASE MONITOR: MONITOR ();
                          BREAK;
            CASE ERROR;   ERROR ();
                          BREAK;
          }
        }
    }

 SEARCH ()
    {
     SEARCH_STATUS = NOT_FOUND;
     WHILE (SEARCH_STATUS = NOT_FOUND)
       {
        FOR (BYTECNT = 0; BYTECNT < 48; BYTECNT=BYTECNT+4)
          {
           IF (NEXT FOUR BYTES HAVE A GOOD CRC AND THE
              NEXT PACKET IS FOLLOWED BY A GOOD CRC)
               {
                SEARCH_STATUS = FOUND;
                STATE = VERIFY;
               }
          }
        IF (LINE_TYPE = UNFRAMED) SHIFT HARDWARE BUFFER
           ONE BIT ();
       }
    }


VERIFY ()
   {
    VERIFY_STATUS = NOT_VERIFIED;
    WHILE (VERIFY_STATUS = NOT_VERIFIED)
      {
       FOR (CRC_CNT = 0; CRC_CNT < 4; CRC_CNT=CRC_CNT+4)
         {
          IF (BAD CRC ON NEXT EXPECTED PACKET BOUNDARY)
            {
             STATE = SEARCH;
             BREAK;
            }
         }
       VERIFY_STATUS = VERIFIED;  /* FOUR GOOD PACKETS */
      }
    IF (STATE = SEARCH) STATE = MONITOR;
   }

 MONITOR ()
    {
     MONITOR_STATUS = CRC_OK;
     WHILE (MONITOR_STATUS = CRC_OK)
       {
        IF (GOOD CRC ON NEXT EXPECTED PKT BOUNDARY)
          {
           IF (LAST PACKET WAS 24 BYTES)
             WRITE LAST PACKET TO MUXBUS ();
          }
        ELSE                          /* BAD CRC */
          {
           MONITOR_STATUS = CRC_BAD;
           STATE = ERROR;
```

```
        }
      }
    }

ERROR ()
  {
   FOR (BYTE_CNT =0; BYTE_CNT < 48; BYTE_CNT=BYTE_CNT+4)
     {
      IF (NEXT 4 BYTES HAVE GOOD CRC)
        {
         IF THIS IS AN EXPECTED BOUNDARY OF EITHER
           A 4 OR 24 BYTE PACKET, WHOSE START WAS FOUND
           IN THE MONITOR STATE OR THIS IS A GOOD CRC
           FOUND IN THIS THE ERROR STATE THEN
           {
            STATE = MONITOR;
           }
        }
     }                                  /* END OF FRAME SYNC */
   STATE = SEARCH;
  }
```

What is claimed is:

1. In a data communication system, a packet boundary detection method comprising the steps of:

transmitting to a destination address a serial bit data stream containing a sequence of data packets, each data packet including a multi-bit error detection value at a predefined position in said data packet relative to said data packet's starting boundary; wherein said multi-bit error detection value in each said data packet is a predefined error code computation function of other bit values in said data packet;

receiving, at said destination address, said serial bit data stream;

finding a packet boundary in said received serial bit data stream by

(A) selecting a first bit position in said received serial bit data stream as a potential packet boundary, and identifying a corresponding potential data packet in said received serial bit data stream;

(B) determining whether data located at said predefined position relative to said potential packet boundary is consistent with a value computed by applying said predefined error code computation function to said identified potential data packet in said received serial bit data stream;

(C) choosing said first bit position as a packet boundary when said determining step finds consistent values, and

(D) otherwise advancing said first bit position by one bit position in said received serial bit data stream and then repeating steps (B) through (D) until a packet boundary is located.

2. The method of claim 1, further including:

after selecting a packet boundary:

(E) finding at least one subsequent packet boundary in said received serial bit data stream consistent with said selected packet boundary, and identifying a potential data packet in said received serial bit data stream corresponding to each said subsequent packet boundary;

(F) determining whether data located at said predefined position relative to each of said at least one subsequent packet boundary is consistent with a value computed by applying said predefined error code computation function to said corresponding identified potential data packets in said received serial bit data stream; and

(G) conforming said selected packet boundary as correct when step (F) finds consistent values, and otherwise advancing said first bit position by one bit position in said received serial bit data stream and then repeating steps (B) through (G) until a packet boundary is located and confirmed.

3. In a data communication system, a packet boundary detection method comprising the steps of:

transmitting to a destination address a serial bit data stream containing a sequence of data packets, each data packet including a multi-bit error detection value at a predefined error code position in said data packet; wherein said multi-bit error detection value in each said data packet is a predefined error code computation function of other bit values in said data packet;

receiving, at said destination address, said serial bit data stream;

finding a packet boundary in said received serial bit data stream by

(A) selecting a set of bits in said received serial bit data stream starting at a first bit position in said received serial bit data stream,

(B) applying said predefined error code computation function to said selected set of bits to compute an error code value,

(C) comparing said computed error code value with a multi-bit data value in said selected set of bits located at a position therein corresponding to said predefined error code position in transmitted data packets;

(D) selecting said first bit position at a packet boundary when said comparing step finds a match; and

(E) when said comparing step does not find a match, advancing said first bit position by one bit position in said received serial bit data stream and then repeating steps (B) through (E) until a packet boundary is located.

4. The method of claim 3, further including:

said selecting a packet boundary:

(F) finding at least one subsequent packet boundary in said received serial bit data stream consistent with said selected packet boundary;

(G) applying said predefined error code computation function to said received serial bit data stream for each said subsequent packet boundary to compute a corresponding error code value;

(H) comparing each said of said computed error code values when a multi-bit data value in said received serial bit data stream located at a position therein corresponding to said subsequent packet boundary and said predefined error code position; and

(I) confirming said selected packet boundary as correct when step (H) finds a match for each said computed error code value corresponding to said at least one subsequent packet boundary, and otherwise advancing said first bit position by one bit position in said received serial bit data stream and then repeating steps (B) through (I) until a packet boundary is located and confirmed.

5. In a data communication system, a packet boundary detection method comprising the steps of:

transmitting to a destination address a serial bit data stream containing a sequence of data packets, each data packet including a multi-byte packet header having a packet destination address and a multi-bit error detection value at a predefined error code position in said multi-byte packet header; wherein said multi-bit error detection value in each said multi-byte packet header is a predefined error code computation function of other bit values in said data packet;

receiving, at said destination address, said serial bit data stream;

finding a packet boundary in said received serial bit data stream by

(A) selecting a set of bits in said received bit data stream starting at a first bit position in said received serial bit data stream,

(B) applying said predefined error code computation function to said selected set of bits to compute an error code value,

(C) comparing said computed error code value with a multi-bit data value in said selected set of bits located at a position therein corresponding to said predefined error code position in said multi-byte packet header;

(D) selecting said first bit position as a packet boundary when said comparing step finds a match; and

(E) when said comparing step does not find a match, advancing said first bit position by one bit position in said received serial bit data stream and then repeating steps (B) through (E) until a packet boundary is located.

6. The method of claim 5, further including:

after selecting a packet boundary:

(F) finding at least one subsequent packet boundary in said received serial bit data stream consistent with said selected packet boundary;

(G) applying said predefined error code computation function to said received serial bit data stream for each said subsequent packet boundary to compute a corresponding error code value;

(H) comparing each said computed error code value with a multi-bit data value in said received serial bit data stream located at a position therein corresponding to said subsequent packet boundary and said predefined error code position; and

(I) confirming said selected packet boundary as correct when step (H) finds a match for each said computed error code values corresponding to said at least one subsequent packet boundary, and otherwise advancing said first bit position by one bit position in said received serial bit data stream and then repeating steps (B)–(I) until a packet boundary is located and confirmed.

7. In a data communication system, a packet boundary detection apparatus comprising:

a data transmitter which transmits to a destination address a serial bit data stream containing a sequence of data packets, each data packet including a multi-bit error detection value at a predefined position in said data packet relative to said data packet's starting boundary; wherein said multi-bit error detection value in each said data packet is a predefined error code computation function of other bit values in said data packet;

a data receiver, at said destination address, which receives said serial bit data stream;

packet boundary detection means for finding a packet boundary in said received serial bit data stream, said packet boundary detection means including:

memory means for storing a portion of said received serial bit data stream; and

computation means, coupled to said memory means, for repeatedly performing a predefined set of computations until a packet boundary is found, including: (A) selecting a first bit position in said received serial bit data stream as a potential packet boundary, and identifying a corresponding potential data packet in said received serial bit data stream; (B) determining whether data located at said predefined position relative to said potential packet boundary is consistent with a value computed by applying said predefined error code computation function to said identified potential data packet in said received serial bit data stream; (C) choosing said first bit position as a packet boundary when said choosing finds consistent values, and (D) advancing said first bit position by one bit position in said received serial bit data stream when said choosing finds inconsistent values and then repeating said predefined set of computations.

8. The apparatus of claim 7 wherein said predefined set of computations performed by said computation means further includes:

(E) finding at least one subsequent packet boundary in said received serial bit data stream consistent with said selected packet boundary, and identifying a potential data packet in said received serial bit data stream corresponding to each said subsequent packet boundary;

(F) deciding whether data located at said predefined position relative to each of said at least one subsequent packet boundary is consistent with a value

computed by applying said predefined error code computation function to said corresponding identified potential data packets in said received serial bit data stream; and

(G) confirming said selected packet boundary as correct when computation (F) finds consistent values, otherwise advancing said first bit position by one bit position in said received serial bit data stream and then repeating said predefined set of computations.

9. In a data communication system, packet boundary detection apparatus comprising:

a data transmitter which transmits to a destination address a serial bit data stream containing a sequence of data packets, each data packet including a multi-bit error detection value at a predefined error code position in said data packet; wherein said multi-bit error detection value in each said data packet is a predefined error code computation function of other bit values in said data packet;

a data receiver, at said destination address, which receives said serial bit data stream;

packet boundary detection means for finding a packet boundary in said received serial bit data stream, said packet boundary detection means including:

memory means for storing a portion of said received serial bit data stream; and

computation means, coupled to said memory means, for repeatedly performing a predefined set of computations until a packet boundary is found, including: (A) applying said predefined error code computation function to a selected set of bits stored in said memory means, starting at a first bit position in said received serial bit stream, to compute an error code value; (B) comparing said computed error code value when a multi-bit data value in said selected set of bits located at a position therein corresponding to said predefined error code position in transmitted data packets; (C) selecting said first bit position as a packet boundary when said comparison means finds a match, and (D) otherwise advancing said first bit position by one bit position in said received serial bit data stream and then repeating said predefined set of computations.

10. The packet boundary detection apparatus of claim 9, said predefined set of computations performed by said computation means further including:

after selecting a packet boundary:

(E) finding at least one subsequent packet boundary in said received serial bit data stream consistent with said selected packet boundary;

(F) applying said predefined error code computation function to said received serial bit data stream for each said subsequent packet boundary to compute a corresponding error code value;

(G) comparing each said computed error code value with a multi-bit data value in said received serial bit data stream located at a position therein corresponding to said subsequent packet boundary and said predefined error code position; and

(H) confirming said selected packet boundary as correct when computation (G) finds a match for each said computed error code value corresponding to said at least one subsequent packet boundary, and otherwise advancing said first bit position by one bit position in said received serial bit data stream and then repeating said predefined set of computations.

11. In a data communication system, a packet boundary detection apparatus comprising:

a data transmitter which transmits to a destination address a serial bit data stream containing a sequence of data packets, each data packet including a multi-byte packet header having a packet destination address and a multi-bit error detection value at a predefined error code position in said multi-byte packet header; wherein said multi-bit error detection value in each said multi-byte packet header is a predefined error code computation function of other bit values in said data packet;

a data receiver, at said destination address, which receives said serial bit data stream;

packet boundary detection means for finding a packet boundary in said received serial bit data stream, said packet boundary detection means including:

memory means for storing a portion of said received serial bit data stream; and

computation means, coupled to said memory means, for repeatedly performing a predefined set of computations until a packet boundary is found, including: (A) selecting a set of bits in said received bit data stream starting at a first bit position in said received serial bit data stream, (B) applying said predefined error code computation function to said selected set of bits to compute an error code value, (C) comparing said computed error code value with a multi-bit data value in said selected set of bits located at a position therein corresponding to said predefined error code position in said multi-byte packet header, (D) selecting said first bit position as a packet boundary when said comparing finds a match, and (E) when said comparing does not find a match, advancing said first bit position by one bit position in said received serial bit data stream and then repeating said predefined set of computations.

12. The apparatus of claim 11, said predefined set of computations performed by said computations means further including:

after selecting a packet boundary:

(F) finding at least one subsequent packet boundary in said received serial bit data stream consistent with said selected packet boundary;

(G) applying said predefined error code computation function to said received serial bit data stream for each said subsequent packet boundary to compute a corresponding error code value;

(H) comparing each said computed error code value with a multi-bit data value in said received serial bit data stream located at a position therein corresponding to said subsequent packet boundary and said predefined error code position; and

(I) confirming said selected packet boundary as correct when computation (H) finds a match for each said computed error code value corresponding to said at least one subsequent packet boundary, and otherwise advancing said first bit position by one bit position in said received serial bit data stream and then repeating said predefined set of computations.

* * * * *